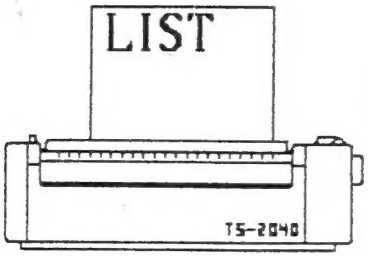
L.I.S.T.ing Newsletter

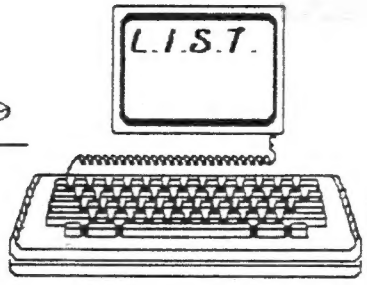
The newsletter of the Long Island Sinclair Timex group.

*** Incorporating N Y T S E ***

HARDWARE PROJECT



Issue: FEBRUARY (MONTH) 1989 (YEAR)

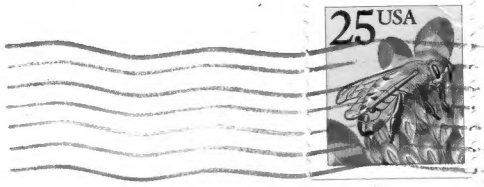


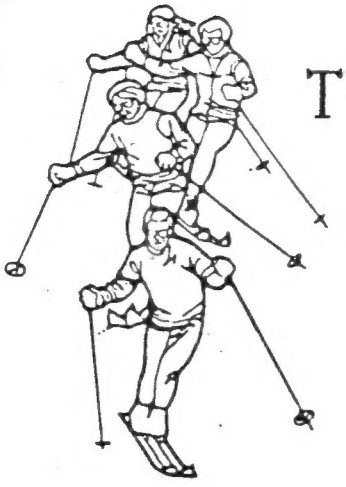
EPROM PROGRAMMER YOU CAN BUILD

L.I.S.T. membership for one year is $15.00. Library tapes are available. Write to the below address for further information.

NEXT MEETING MARCH 12, 1989

L.I.S.T.
5 Peri Lane
Valley Stream, NY 11581





TO:

Don Lambert JAN/90
3310 Clover Dr. S
Cedar Rapids, IA
52404

FIRST CLASS MAIL
DATED MEETING NOTICE
Please DON'T delay!!!

LIST OFFICERS

EDIT. HARVEY RAIT
TREA. ROBERT MOLLOY
EDITOR. STEVE KAYE
EDITOR. FRED STERN
LIBR. TOM SKAPINSKI

PLEASE SEND SUBMISSIONS OR INQUIRIES TO

LIST
MR. HARVEY RAIT
5 BERRI LANE
VALLEY STREAM, N.Y., 11581

COMING EVENTS:

MARCH 18, 1988 LIST MEETING
MARCH 13, 1988 NYTSE MEETING

MEETING MINUTES
FEBRUARY 18, 1988

THE MEETING STARTED AT 2:15PM
HARVEY ANNOUNCED THAT THE MEETING WOULD BE HELD FIRST FOLLOWED BY THE AUCTION.
ALSO THE ARCADE CONTROLLER, P.S. AND LISP DRIVES WOULD NOT BE ALLOTTED.
THE SOLDER WILL BE ON THE DISC DRIVES AND THE PACKAGE WILL BE ALLOTTED AT A FUTURE DATE.

FRED STERN ANNOUNCED
LETTERS WERE SENT TO VARIOUS COMPUTER NEWSPAPERS ASKING TO PLACE IN THEM THE LIST MEMBERSHIP ADVERTISEMENT.
ONLY YANKEE TRADER ACKNOWLEDGED OUR REQUEST.
MICRODIGITAL ELECTRONICS IN ESPAÑA WAS ASKED TO SEND INFORMATION ABOUT THEIR COMPUTERS.
TIME DESIGNS HAD AN ARTICLE WHICH STATED THEY HAD A COMPUTER WHICH WAS COMPATABLE WITH THE TS2068.
FINALLY, A LETTER WAS SENT TO TIMEX/SINCLAIR PUBLIC DOMAIN SOFTWARE REQUESTING INFORMATION ABOUT THEIR PRODUCTS.

GREG PAPOVICH TOLD US OF A TIMEX/SINCLAIR NEWSLETTER WHICH IS TO BE PUBLISHED ON FIDO NET. THIS NEWSLETTER WILL BE FOR ALL MACHINES (TS1000, TS2068, QL, AND Z88).
GREG ALSO TOLD US OF SOFTWARE HE WAS DEVELOPING FOR USE WITH THE LARKEN DOS.
CONTACT GREG FOR MORE DETAILED INFORMATION ON HIS BBS 201-956-7703.

LISTING NEEDS YOUR SUPPORT

ONCE AGAIN I AM REQUESTING FROM YOU, OUR READERS, SUPPORTERS, AND SUBSCRIBERS TO DONATE ARTICLES OF INTERESTFOR REPRINT IN LISTING. YOUR ARTICLE CAN EITHER BE TYPED OR HAND WRITTEN AS LONG AS IT IS IN ENGLISH, LEGIBLE, AND PERTAINENT TO TIMEX/SINCLAIR COMPUTERS OR PERIPHERALS.

JOHN FACMINI ANNOUNCED
SHARP WILL BE SELLING AN IBM PC AND MSDOS EMULATOR FOR THE QL. THIS EMULATOR WILL ALLOW YOU TO RUN PC PROGRAMS ON YOUR QL.
THE EMULATOR WILL BE AVAILABLE ON DISC IN TWO FORMS.
(1) IBM PC EMULATOR ONLY
(2) IBM PC EMULATOR AND MSDOS VERSION 4.0.
CAPITAL AREA TIMEX SINCLAIR COMPUTERFEST WILL BE HELD MAY 6/7 AT THE HOWARD JOHNSONS PLAZA HOTEL NEW CARROLLTON MD.
SINCLAIR AT NIGHT SEES IT ON LINE AND CAN BE CONTACTED AFTER 11:00PM 718-827-1333(8/N/1)
JOHN FINALLY REPORTED ON A VISIT HE MADE TO ZEBRA SYSTEMS. STUART DONATED A BOX OF ASSORTED PROGRAM TAPES TO THE LIST LIBRARY. STUART INFORMED JOHN THAT HE WAS DEFINITELY GETTING OUT OF THE TIMEX TS2068 BUSINESS AND THAT HE WAS LOOKING FOR SOMEONE TO BUY HIS REMAINING STOCK.

THE AUCTION HELD AFTER THE MEETING WAS A BIG SUCCESS. SOME OF THE BARGAINS WERE;
(2) TS1000 FOR $10.00
VARIOUS BOOKS FOR $1.00 EACH
TS2068 TECHNICAL MANUAL $8.00
64K RAM PACK $10.00
ZENITH AMBER MONIT $85.00
THIS IS THE FIRST OF HOPEFULLY MANY AUCTIONS. THE MONEY MADE HELPS KEEP DUES AT ITS CURRENT $15.00 PER YEAR RATE AND IS USED TO PAY FOR CLUB EXPENSES SUCH AS PRINTING AND POSTAGE OF THE NEWSLETTER, PURCHASING TAPES TO BE USED FOR LIST TAPE NO. 10, AND PURCHASING ITEMS FOR THE NEXT AUCTION.

CLASSIFIEDS

THIS CLASSIFIED SECTION IS AVAILABLE TO ALL LIST MEMBERS FREE OF CHARGE.
THE ONLY RESTRICTION IS THAT IT IS TO BE USED ONLY FOR THE SEEKING, SELLING OR SWAPPING OF SINCLAIR, TIMEX OR MICRODACE COMPUTER EQUIPMENT, PERIPHERALS AND SOFTWARE.
LISTING, LIST, AND ITS OFFICERS DO NOT ENDORSE, WARRANTY, OR GUARANTEE ANY OF THE ITEMS LISTED IN THIS CLASSIFIED SECTION

I AM SEEKING 2 OR MORE HOME ROM CHIPS, P/N 8I88SO, U18 IN THE TS2068. CALL FRED STERN 518-737-0983.

A FINAL WORD :

MY NAME IS FRED STERN, AND I AM THE EDITOR OF THIS EDITON OF LISTING.
ONCE AGAIN I THANK TOM SKAPINSKI AND STEVE KAYE FOR THEIR ASSISTANCE IN GETTING THIS NEWSLETTER TOGETHER.
THIS MONTHS FEATURE ARTICLE IS AN EPROM PROGRAMER WHICH CAN BE USED WITH EITHER TS2068 OR TS1000.
TS2068 ROM BYPASS BY WILLIAM PEDERSEN WILL CONTINUE NEXT MONTH.

HARDWARE PROJECT: EPROM Programmer You can Build!

The cartridge dock of the 2068 was supposed to open up a whole new dimension in instant loading software when Timex introduced it oh so long ago. There were very few cartridges available and what was wasn't worth it. Then came along Doug Dewey's Spectrum emulator which made excellent use of it. Now more recently, OS-64 from Zebra demonstrates another good example.

The cartridge dock supports LROS (Language ROM Oriented Software) and AROS (Application ROM Oriented Software). The Spectrum emulator and OS-64 are both good examples of LROS and all the programs such as those put out by Timex, Sunset Electronics, and Tom Woods' ProFile on cartridge are all good examples of AROS.

With this EPROM programmer, you can develop your own LROS and AROS cartridges. The EPROMs will plug into Doug Dewey's EMU series, Sunset's EPROM boards, and Foote Software's Printer interface EPROM socket.

The programmer will program 2764 (8Kx8) and 27128 (16Kx8) EPROMs and with some slight modification, will program the 2716, 2732, and the 27256 EPROMs. The 2068 and Spectrum ROMs are the 27128 configuration and the 2068 extension ROM is configured as a 2764.

You need to use 250 or 300nS type EPROMs which are designated as 27128-25 or 27128-3. This is the highest speed at which they will operate and must be able to operate faster than the 2068 will access them so they can keep up!

To program these particular EPROMs, you need a 21 volt programming voltage. The 2068's power supply is rated at 15 volts with a one amp load. With nothing connected, it measures 22-23 volts. With it loaded down by a 2068 and a printer interface, disk drive and this interface, it measures between 20 and 20.5 volts. I have been using this voltage to program EPROMs and have had no trouble.

I checked a number of 2068s and power supplies to see if these findings were consistent, they were, so the 2068's power supply should work well for you too. If for some reason it does not, the TS2040's power supply and a regulator circuit will work fine.

Programming the EPROM is a fairly simple task of providing the 21 volt programming voltage, setting the address lines to the desired address you wish to program data to, placing data on the data lines and then making the program line go low for about 50 mS (1/20 sec).

With this interface, you can set the starting address of the EPROM where you want the programming to begin, set the starting address in the computer's memory you wish to program from and set the number of locations you wish to program. A very simple system that will allow you to use this interface with any Sinclair computer or other computers for that matter as long as the correct programming voltage is supplied to the EPROM.

The heart of the programmer is the decoder which creates the ports from which control to different parts of the programmer are obtained. I chose a 74LS138 3 to 8 line decoder which I used to fully decode I/O ports 0-7. Not all the ports are needed, so a few can be used for another project later. I chose the following ports for this project:

PORT 0: READ EPROM
PORT 2: LOW BYTE ADDRESS
PORT 3: HIGH BYTE ADDRESS
PORT 4: PROGRAM EPROM
PORT 5: INCREMENT ADDRESS

Because we are programming a 14 bit EPROM from an 8 bit source, we must load the EPROM starting address in two steps. First the lower 8 bits and then the upper 6 bits. If we want to start at say address 10,000 decimal (2710 hex), we need to load the high byte as OUT 3,39 (27h) and the low byte as OUT 2,16 (10h). How do we come up with those numbers? High byte = INT(10,000/256)=39 & Low byte = 10,000-256*INT(10,000/256)=16. You can convert these numbers to hex by looking in appendix B on page 239 in the 2068 User's manual.

From the schematic, we can see that these low and high bytes are loaded into 74LS193 presettable counters which can then be incremented by sending an OUT command to port 5. Port 0 is used with the IN command to read the EPROM and port 4 is used with the OUT command to write data to the specified address.

The 74LS245 (a) is used primarily as a buffer to keep from loading down the data buss of the 2068. The 74LS245 (b) is used to isolate latched program data from the buss. The 74LS374 is used to latch or hold data to be programmed. The 74LS221 is used as a pulse generator to give the 50 mSec pulse needed by the EPROM during programming. The 74LS27 and 74LS08 are used in conjunction with the 74LS138 to give full decoding of the I/O ports.

The programming voltage from the 2068 goes through a resistor and 22 volt zener diode to help protect both the 2068 and EPROM. A 0.1 microFarad tantalum capacitor helps cut down on transient voltages and five volts feeds the line through a diode so that the EPROM can be read when not in the program mode. I tried lowering the program voltage some to see how much effect it would have and found that I could go as low as 18 volts and still have reliable programming. This may vary from manufacturer to manufacturer though. I have been using Hitachi # HN4827128G-25.

The Zebra Systems Protocard makes an excellent board to mount this project on. I used a proto board to develop the programmer and later transfered it to a Protocard. Keep the leads as short as possible when wiring the circuit and be carefull not to cause any shorts, solder bridges, frayed wires, etc. Remember, this project deals with the data and address buss of the computer and mistakes can easily ruin the computer.

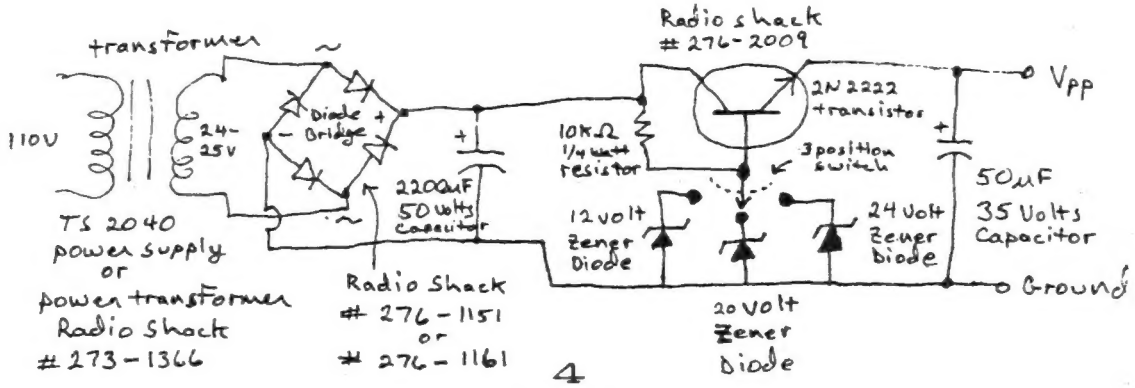
To program the 2716, you need a 25 volt power supply and to program the 27256, you need 12.5 volts. Here is a wiring diagram to use the TS 2040 printer power supply or any 25 volt secondary transformer for the programming supply.

The 2716 and 2732 EPROMs are in a 24 pin package with pins 1-12 & 13-24 corresponding to pins 3-14 & 15-26 of a 28 pin package. The Vcc is moved from pin 28 to 24. All the pins remain pretty much the same except for the highest address line and PGM which move some from EPROM to EPROM depending on its size. These changes can be realized by using wire jumpers or switches.

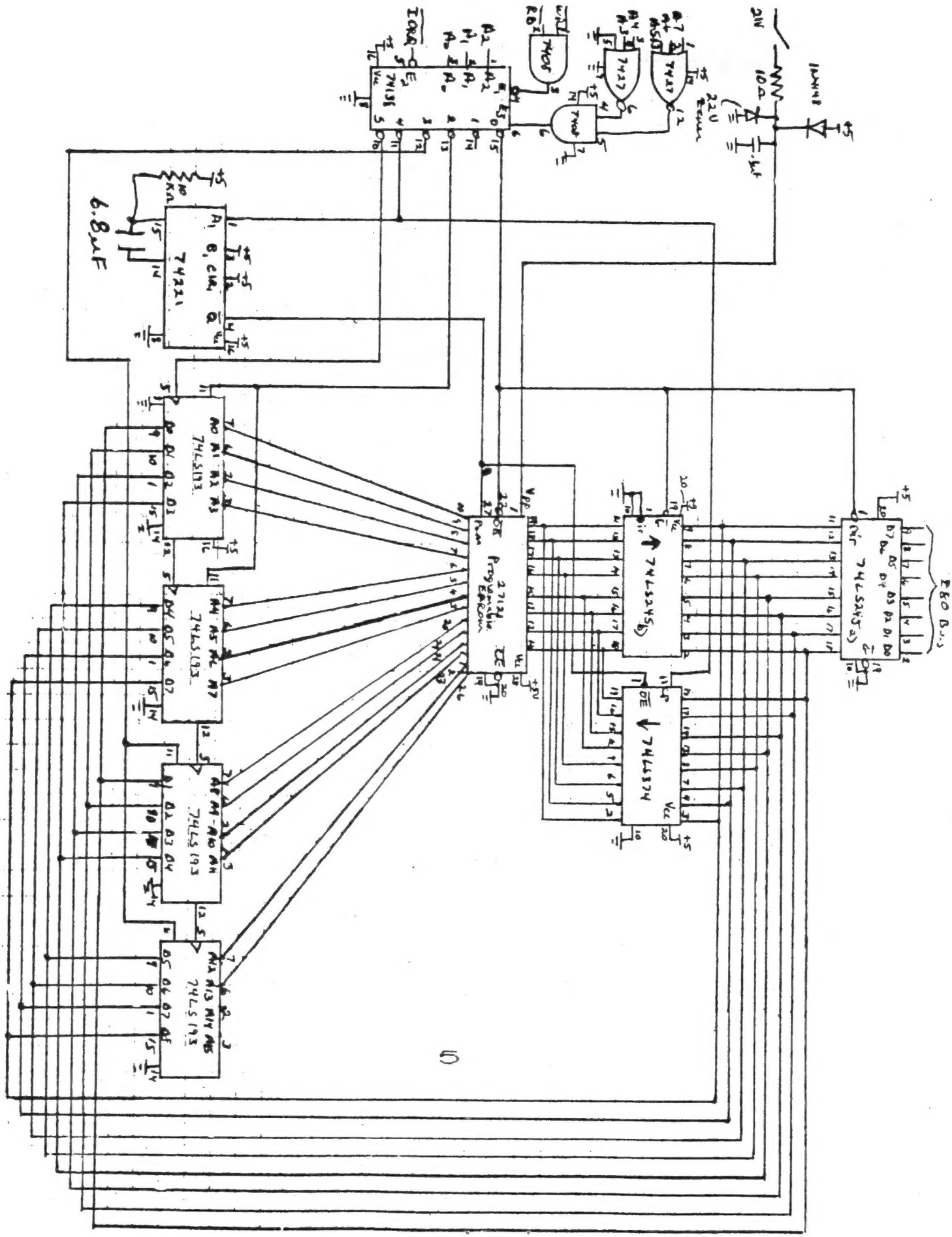
Because the 2068 uses ROMs, some of the pins are slightly different. To use an EPROM in place of the ROM, pins 1 & 27 of the EPROM need to be bent upward (away from socket) and connected together and connected to pin 28 so that they are all connected to Vcc (5 volts).

Once you have completed the interface, you need to learn how to get the most from it. There are some good books and programs out there that would be most helpful with this programmer. For machine code writing and de-bugging, Hot-Z by Ray Kingsley, PO Box 8032, Santa FE, NM 87504 is a must. It is available on tape or cartridge. Sunset Electronics is selling a 2068 ROM disassembly manual which also covers programming your own EPROMs, and John Oliger also has plans and complete EPROM programers also. His address is: 11601 Whidbey dr, Cumberland IN 46229.

Here is a simple little BASIC program to program the 2764 and 27128 EPROMs with the 2068. Next time I will show you how to use this interface with the TS 1000/1500. Also, I'll discuss correcting some of the bugs in the 2068 home ROM, and



EPROM Programmer Schematic


74LS193
74LS245
74LS374
2716
EPROM Programmer
74221
6.8uF
1.5uF
220 Ohms
100
74LS138
74LS00
74LS27
74LS32
74LS08
74LS86
74LS04
74LS157
74LS244
74LS273
74LS374
74LS245
74LS193
74221
2716
EPROM Programmer
74LS193
74LS245
74LS374

transferring your own programs to EPROM.

```
   5 REM EPROM Programmer
  10 REM Read EPROM
  15 INPUT "enter 0 to read EPRO
M                enter 1 to write to
EPROM: ";type
  18 IF type THEN GO TO 100
  20 INPUT "enter start address
of ROM(d) ";s
  30 GO SUB 1000
  40 INPUT "enter length of code
(d) ";len
  50 FOR n=0 TO len-1
  60 PRINT s+n,IN 0
  70 OUT 5,0
  80 NEXT n
  90 GO TO 15
 100 REM Programming the EPROM
 115 INPUT "enter memory start l
oc(d) ";m: LET mv=m
 120 INPUT "enter start address
of ROM(d) ";s
 130 GO SUB 1000
 140 INPUT "enter length of code
(d) ";len
 150 FOR n=0 TO len-1
 165 LET d=PEEK m: LET m=m+1
 170 OUT 4,d
 175 PAUSE 4
 180 OUT 5,0
 190 NEXT n
 195 PRINT "programmed"''
 200 REM Verify
 205 GO SUB 1000
 210 FOR n=0 TO len-1
 230 LET d=PEEK mv: LET mv=mv+1
 240 IF IN 0<>d THEN PRINT s+n:
STOP
 245 OUT 5,0
 250 NEXT n
 260 PRINT "verified"
 999 STOP
1000 OUT 2,INT (s-256*INT (s/256
))
1010 OUT 3,INT (s/256)
1020 RETURN
```

The Unprintable Characters

The purpose of this article is to illustrate the power of the CHR$ function. The argument for the CHR$ function is a numeric value code from 0 to 255. This is the entire character set of single letter characters, keywords, and tokens. Refer at this point to the character set listing in the appendix of your programming manual. You will see that character codes 67-111, 122-125, and 195 are not used in the ZX81 system. You can verify this by running the program in Figure 1. The unused codes are shown as question marks.

Figure 1.

```
10 FOR N=0 TO 255
20 PRINT CHR$ N;
30 NEXT N
```

Now change line 10 to 10 FOR N= 128 to 191 and hit RUN then ENTER. The characters displayed cannot be entered from the keyboard itself. The interesting thing about this concept is that graphic strings are printed very rapidly and this may be applied to any work dealing with high speed graphics.

The CHR$ function can also be used to compare other nonprintable functions such as the cursor operation symbols like "◆"; "◆"; "◆"; "◆"; which cannot be placed onto the screen. This is accomplished by applying a string equivalent, which can be called by using their respective CHR$ codes (112-115). We can examine an example of that code by using the CHR$ (114) for the backspace cursor in Figure 2. Type in Figure 2 at this point.

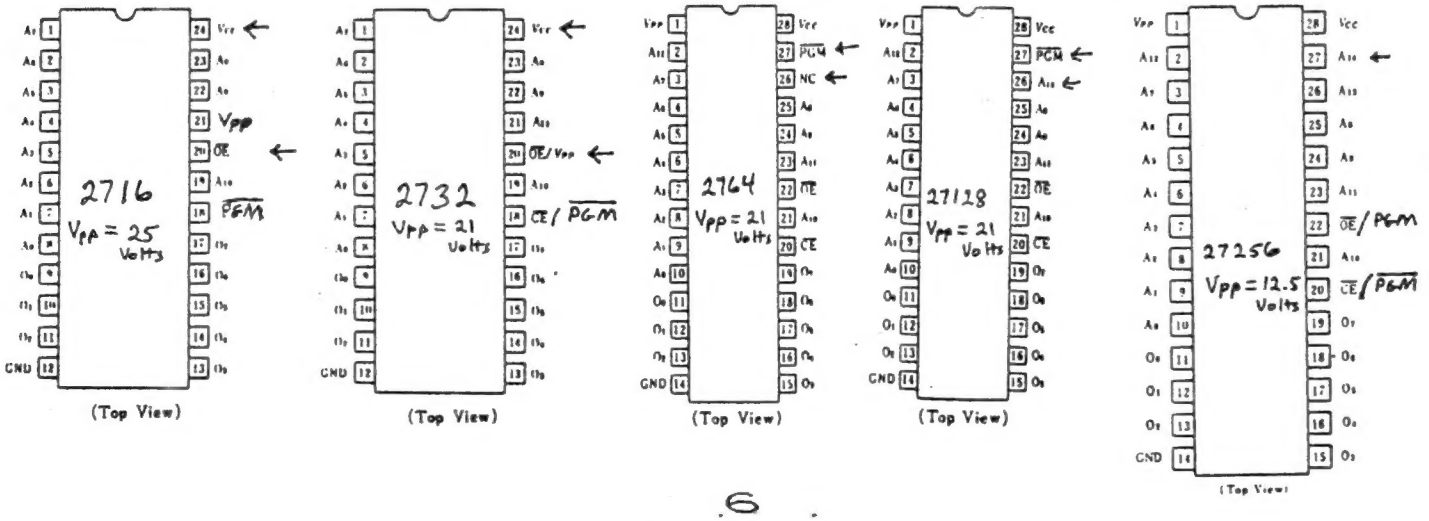
Figure 2.

```
 10 LET A$=INKEY$
 20 IF A$="" THEN GOTO 10
 30 IF A$<>CHR$ 114 THEN GOTO 7
0
 40 SCROLL
 50 PRINT "LEFT CURSOR ON"
 60 GOTO 10
 70 SCROLL
 80 PRINT A$
 90 GOTO 10
```

In the SLOW mode hit RUN then ENTER. What happens when you depress SHIFT then hit 5? Try the other SHIFTed cursor positions. Experiment with ENTER and other keyboard letters, numbers, and tokens. Hit BREAK STOP the program when you are finished.

Various EPROM pin configurations of most popular EPROMs

note differences & simularities between pins as memory size changes



EPROM Programmer You Can Build, Part II

To use the EPROM programmer discussed last month with the TS 1000/1500, you need to use the program listed here. Because the 1000/1500 does not have IN or OUT, a couple of short machine code routines appear in the REM statement of line 1.

Memory location 16514 holds data to be outputted, and 16515 holds incomming data. 16521 is POKEd with the correct port to output from. The OUTput routine starts at 16516 and the INput routine starts at 16524.

Here is the machine code listing using decimal addresses:

```
16514                          ;data
16515                          ;data
16516 F5      push af          ;save registers
16517 3A8240  ld a,(16514)     ;ld data in reg.
16520 D3N?    out (N),a        a, out to port n
16522 F1      pop af           ;restore reg's.
16523 C9      ret              ;return to prog.
16524 F5      push af          ;
16525 DB00    in a,0           ;in data from 0
16527 328340  ld (16515),a     port, ld data to
16530 F1      pop af           16515
16531 C9      ret              ;return to prog.
```

The first time you LOAD the program, use GOTO 270 to place the machine code in the REM statement in line 1. Then delete lines 270 onward and save this version.

The best use of this EPROM programmer is to have some type of memory board such as Hunter's Non-volatile memory board for the 1000/1500 which occupies the 8-16K area. You could even make some cartridges for the few TS 1510 cartridge players that exist.

The 1500 can auto-RUN programs stored on cartridge provided that memory location 8192d is set (=1) and starts executing at 8193. The 2068 has a much more involved way of setting up for cartridge use. The first eight bytes of the cartridge or EPROM contain information as to the type of program and where it is banked to and where the program starts. On power up, the computer checks these eight bytes and acts accordingly.

```
  1 REM DO PRINT USR PEEK 16514 =LET
 TAN PRINT (= M=RND LET TAN
  5 REM ***EPROM PROGRAMMER***
 10 REM ***READ EPROM***
 15 PRINT "ENTER 0 TO READ EPRO
M                ENTER 1 TO WRITE TO
EPROM: "
 16 INPUT TYPE
 18 IF TYPE THEN GO TO 100
 20 PRINT "ENTER START ADDRESS(
DEC)"
 25 INPUT S
 30 GO SUB 270
 40 PRINT "ENTER LENGTH OF CODE
(DEC)"
 45 INPUT LEN
 50 FOR N=0 TO LEN-1
 55 RANDOMIZE USR 16524
 60 PRINT S+N,PEEK 16515
 70 POKE 16521,5
 75 RANDOMIZE USR 16516
 80 NEXT N
 90 GO TO 15
100 REM PROGRAMMING THE EPROM
110 PRINT "ENTER MEMORY START L
OCATION(DEC)"
115 INPUT M
118 LET MV=M
120 PRINT "ENTER START ADDRESS(
DEC)"
125 INPUT S
130 GO SUB 270
140 PRINT "ENTER LENGTH OF CODE
(DEC)"
145 INPUT LEN
150 FOR N=0 TO LEN-1
155 POKE 16514,PEEK M
156 LET M=M+1
170 POKE 16521,4
172 RANDOMIZE USR 16516
173 PAUSE 1
180 POKE 16521,5
185 RANDOMIZE USR 16516
190 NEXT N
195 PRINT ,,"PROGRAMMED",,,,
200 REM ***VERIFY***
205 GO TO 270
210 FOR N=0 TO LEN-1
230 LET D=PEEK MV
232 LET MV=MV+1
237 RANDOMIZE USR 16524
240 IF PEEK 16515<>D THEN PRINT
S+N
245 POKE 16521,5
247 RANDOMIZE USR 16516
250 NEXT N
260 PRINT ,,"VERIFIED"
265 STOP
270 POKE 16521,2
272 POKE 16514,INT (S-256*INT (
S/256))
274 RANDOMIZE USR 16516
280 POKE 16521,3
285 POKE 16514,INT (S/256)
290 RANDOMIZE USR 16516
295 RETURN
300 LET A$="00000024F0581300648
1100324180124521900005013108424 1
801"
301 LET M=16514
302 FOR N=1 TO 54 STEP 3
303 POKE M,VAL A$(N TO N+2)
304 LET M=M+1
305 NEXT N
```

The first byte tells the computer whether the program is in BASIC (01) or machine code (02). The second byte tells whether it is AROS (01) or LROS (02). The third and fourth tell where the starting address is. The fifth tells what "chunk" of memory the program resides in, the sixth tells whether it is auto start (01) or not. The seventh and eighth tells the number of variables being used plus 21.

The "chunk" byte can be in 8 or 16K blocks and set using the following codes:

Chunk	Addresses	Decimal code 8K	16K
0	0-8191	1	3
1	8192-16383	2	
2	16384-24575	4	12
3	24576-32767	8	
4	32768-40959	16	48
5	40960-49151	32	
6	49152-57343	64	192
7	57344-65535	128	

To see how this works, lets look at the first ten bytes in three of the most popular cartridge software: The Spectrum Emulator, OS-64, and Hot-Z.

	Spectrum Emulator	OS-64	Hot-Z
0	243	0	2
1	1	1	2
2	110	5	8
3	56	0	128
4	252	252	207
5	195	195	1
6	203	158	21
7	17	13	0
8	42	42	251
9	93	93	205

The Spectrum Emulator and OS-64 are both LROS type as designated by the 1 in the second byte and Hot-Z is an AROS type as designated by the 2. For LROS type, it does not matter what the first byte is, so for the emulator and OS-64, the numbers there are insignificant. The 2 in Hot-Z means that it is machine code.

The Spectrum emulator starts executing at 256*56+110=14446 due to the numbers in the third and fourth byte. OS-64 starts at 256*0+5=5, and Hot-Z starts at 256*128+8=32776. From the fifth byte and the table above, we can see that Hot-Z is banked into the 32768-49151 region, and both OS-64 and the Spectrum emulator are in the 0-16383 region.

The rest of the bytes are not used at startup and can be used for part of the program for LROS, but for Hot-Z, which is AROS, we can see that it is set for auto-run by the 1 in the sixth byte.

For a BASIC program, the first byte should be 01, the second byte should be a two, and bytes 3 & 4 should be the start address of where the BASIC program begins. The sixth byte should be 01 for auto-RUN. It would be best to start an AROS program at 32768 as Hot-Z does to avoid any unwanted confrontations of the system variables which usually end around 26700.

Variables must be initialized in the BASIC program and DEF FN and FN do not work from cartridge. For more information on making and using cartridges, contact: Bob Orrfelt of GESSO Products, 3436 Bay Road, Redwood City, CA 94063.

The Technical Manual for the 2068 (sold by Time Designs Magazine) has some of the code for the 2068 home ROM to correct that obnoxious stop-before-delete bug, the INT-65535 bug and others. To make corrections to the home ROM, you need to copy the code up into memory so that you can make the changes. To stick with even numbers, start the code at 40,000 by first entering CLEAR 39999:NEW and then the following:

```
2000 FOR N=0 TO 16383: POKE N+40
000,PEEK N: NEXT N
```

Now the code is up where you can manipulate it. Make the following POKEs to correct the listed bugs:

DELETE delay	INT-65536 etc. cont.
40849,1	43794,24
40850,1	43795,26
40851,0	43796,241
40852,11	43797,119
40853,121	43798,35
40854,176	43799,114
40855,32	43800,35
40856,251	43801,113
40857,241	43802,43
40858,24	43803,43
40859,210	43804,43
	43805,209
Optional turn on	43806,201
message: (Use code	43807,241
for each char.)	43808,43
44376 Good Day!	43809,54
44386 Revision 2.1	43810,145

44400 July 1986...	43811,35
add 128 to last	43812,54
character	43813,128
	43814,60
INT-65536 etc.	43815,24
53297,245	43816,237
53298,60	43817,255
53299,179	43818,255
53300,178	43819,255
53301,194	43820,255
53302,228	43821,255
53303,53	
53304,195	
53305,239	
53306,53	

After you have made all the corrections you wish to make, use the program shown last time to program your EPROM. When the program asks for memory start loacation, enter 40000. Answer 0 to "Enter start address of ROM and 16384 for length of code question. The EPROM will take about 12 minutes to program and will verify afterwards. If all goes well, you will have your own personalized 2068 operating system when done.

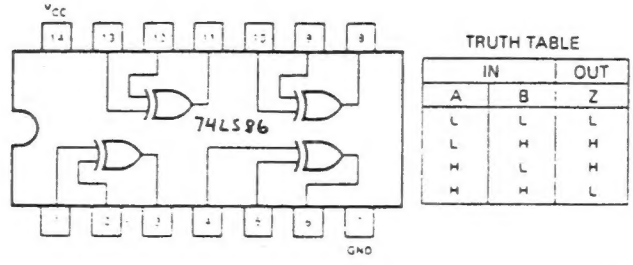
For those with QL's that have started to tinker around with your own hardware, you should find it fairly easy to convert this programmer to make your own EPROM based software cartridges for it as well.

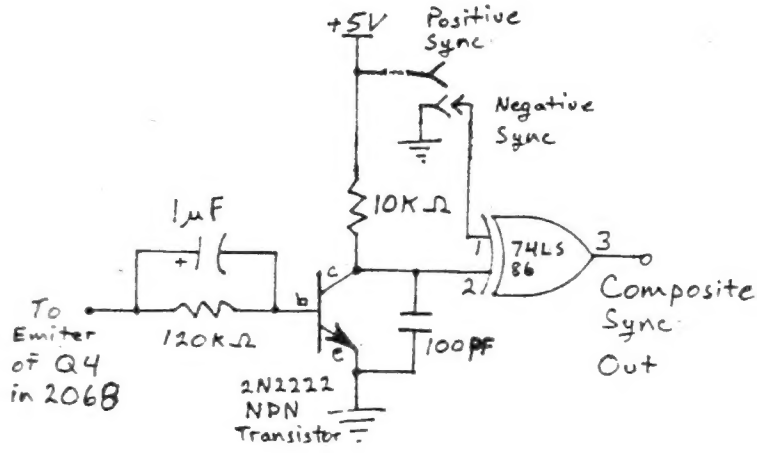
The possibilities are endless with what you can do with the cartridge ports and this EPROM programmer. Enhanced operating systems, utilities, quick load programs with memory saving techniques are just a few of ideas. How about both an enhanced 2068 operating system and a Spectrum operating system all on one 32K EPROM?

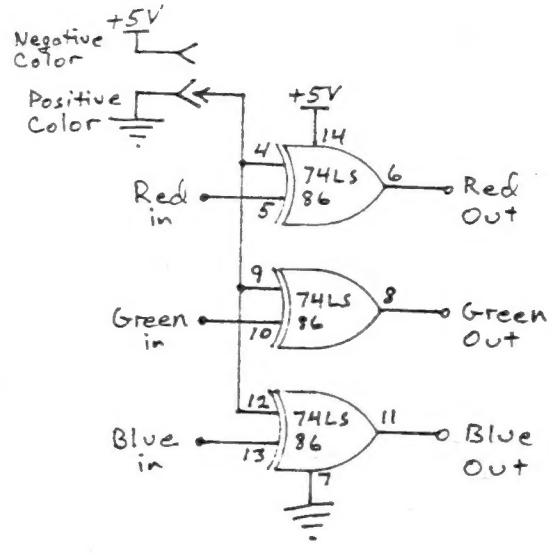
One final note, after you program a few of these EPROMs, you may find that you will want to erase some. They require ultra-violet light for erasure, i.e., a special lamp is needed to erase them. The commercial erasers run on up into the hundreds of dollars. One eraser that I have found to be very cheap and effective is the DATARASE by Walling Co. available from R&D Electronic Supply, 100 E. Orangethorpe Ave., Anaheim, CA 92801 (714) 773-0240. They take plastic and phone orders. Price was $34.95 when I got mine and it will erase two EPROMs in about 8-10 mins.

Joe williamson

Better RGB Circuit







The circuit described in the february 1986 issue of SUM was designed around the Sears RGB/TV/Monitor (same as the Sanyo model # 31C426) and the Magnavox (NAP) RGB-40 or 80.

Both of these monitors (and the QL monitor as well) take negative-going composite sync and active high RGB signals and have well buffered inputs to clean up any poor inputs.

Calculating the Days

Ronald Paludan

Knowing the number of days between two dates is frequently necessary for calculating interest, apportioning expenditures, or calculating average use of items. We are also interested in the more trivia type questions such as how many days before an important date, how many days between certain historical dates, and even how many days we have lived.

This program will enable you to determine the number of days between dates either for serious use or just curiosity. The program in Listing 1 is for the 8K ROM machine with 1K RAM. A 4K ROM adaptation is included in Listing 2.

Operating Instructions

1) Type in the program in Listing 1. Hit RUN and ENTER.

2) The prompts will ask the user to ENTER FIRST DATE. Do so in this order: Type in the month as a number from 1 to 12 and hit ENTER. Type in the day of the month and ENTER. Finally, type in the year giving all four digits and ENTER.

3) Repeat this process for the second date. The first date must be earlier than the second. On the 4K ROM the two dates cannot be more than 88 years apart, but a method is supplied for dealing with this problem.

4) The answer will then be given on the screen.

5) To repeat, hit ENTER. To quit, hit any number or letter and then ENTER.

Error Checking

The input routine does not allow the user to enter more than 12 for a month or 31 for a day. The user with enough memory may want to add a more detailed error checking routine that rejects nonexistent dates, e.g., June 31.

Modifications for the 4K ROM

Listing 2 gives the lines which must be altered to allow the program to operate properly on the 4K ROM (ZX80 or MicroAce).

4K ROM Problem: Dates more than 88 Days apart

The integer Basic of the 4K ROM ZX80 will not handle numbers greater than 32767. For this reason, the program will reject the second year if it is more than 88 years after the first. However, there is a method for finding the number of days between two dates more than 88 years apart. This method requires the use of one or more mid-date(s) which must be less than 89 years from the date next to it in the series. The number of days between the first date and the mid-date is computed and the results recorded. Then the number of days between the mid-date and the second date is computed and added manually to the first result.

Listing 1.
Days between Two Dates (8K ROM).

```
  20 DIM N(12)
  25 LET N(1)=0
  30 LET N(2)=31
  35 LET N(3)=59
  40 LET N(4)=90
  45 LET N(5)=120
  50 LET N(6)=151
  55 LET N(7)=181
  60 LET N(8)=212
  65 LET N(9)=243
  70 LET N(10)=273
  75 LET N(11)=304
  80 LET N(12)=334
  85 LET D=0
  90 PRINT "                              M
/D/Y"
 100 PRINT "ENTER FIRST DATE: ";
 110 INPUT M1
 112 IF M1>12 THEN GOTO 110
 120 PRINT M1;"/";
 130 INPUT D1
 132 IF D1>31 THEN GOTO 130
 140 PRINT D1;"/";
 150 INPUT Y1
 160 PRINT Y1
 170 PRINT "ENTER SECOND DATE: "
;
 180 INPUT M2
 185 IF M2>12 THEN GOTO 180
 190 PRINT M2;"/";
 200 INPUT D2
 205 IF D2>31 THEN GOTO 200
 210 PRINT D2;"/";
 220 INPUT Y2
 225 IF Y2-Y1>88 OR Y2<Y1 THEN G
OTO 220
 230 PRINT Y2
 240 FOR X=Y1 TO Y2-1
 250 LET D=D+365
 255 IF INT (X/100)*100=X AND IN
T (X/400)*400<X THEN GOTO 265
 260 IF INT (X/4)*4=X THEN LET D
=D+1
 265 NEXT X
 270 IF Y1=Y2 THEN LET D=365-D
 275 IF INT (Y1/100)*100=Y1 AND
INT (Y1/400)*400<Y1 THEN GOTO 28
5
 280 IF INT (Y1/4)*4=Y1 AND M1>2
 THEN LET D=D-1
 285 IF INT (Y2/100)*100=Y2 AND
INT (Y2/400)*400<Y2 THEN GOTO 30
0
 290 IF INT (Y2/4)*4=Y2 AND M2>2
 THEN LET D=D+1
 300 LET D=D-N(M1)-D1
 310 LET D=D+N(M2)+D2
 315 IF Y1=Y2 THEN LET D=D-365
 320 PRINT D;" DAYS APART"
 330 INPUT Y$
 340 CLS
 350 IF Y$="" THEN GOTO 20
```

Listing 2. 4K ROM Adaptations.

```
 255 IF 100*(X/100)=X AND 400*(X
/400)<X THEN GO TO 265
 260 IF 4*(X/4)=X THEN LET D=D+1
 270 IF Y1=Y2 THEN LET D=D-365
 275 IF 100*(Y1/100)=Y1 AND 400*
(Y1/400)<Y1 THEN GO TO 285
 280 IF 4*(Y1/4)=Y1 AND M1>2 THE
N LET D=D-1
 285 IF 100*(Y2/100)=Y2 AND 400*
(Y2/400)<Y2 THEN GO TO 300
 290 IF 4*(Y2/4)=Y2 AND M2>2 THE
N LET D=D+1

 315   Delete
```

As an example of how this method works, let's calculate how many days apart July 7, 1881, and January 1, 1981 are.

```
ENTER FIRST DATE:  7/7/1881
ENTER SECOND DATE: 1/1/1921
  (mid-date)
  14423 DAYS APART

    (ENTER)

ENTER FIRST DATE:  1/1/1921
  (mid-date)
ENTER SECOND DATE: 1/1/1981
  21915 DAYS APART

    14423
  + 21915
  -------
    36338 Days apart
```

Variables Used

N(n)	Days used in year by nth month
D	Day counter (Number of days between dates)
M1	Month of 1st date
D1	Day of month for 1st date
Y1	Year of 1st date
M2	Month of 2nd date
D2	Day of month for 2nd date
Y2	Year of 2nd date
X	Temporary variable used as counter

Routines

The routines are found at the following lines:

20-85: Initialization.

95-230: Input dates.

240-310: Count days between dates. Compensate for leap years.

320: Print results.

330-340: Input ""to repeat.

Examples

The American Civil War lasted from February 8, 1861, to May 26, 1865. How many days did it last?

```
                          M/D/Y
ENTER FIRST DATE:  2/8/1861
ENTER SECOND DATE: 5/26/1865
1568 DAYS APART
```

John has a report due on July 20, 1982, and today is May 15, 1982. How many days does he have to complete the report?

```
                          M/D/Y
ENTER FIRST DATE:  5/15/1982
ENTER SECOND DATE: 7/20/1982
66 DAYS APART
```

Herb was born on August 1, 1945. How many days old will he be on his 40th birthday?

```
                          M/D/Y
ENTER FIRST DATE:  8/1/1945
ENTER SECOND DATE: 8/1/1985
14610 DAYS APART
```

```
                          M/D/Y
ENTER FIRST DATE:  1/1/1980
ENTER SECOND DATE: 4/1/1980
91 DAYS APART


                          M/D/Y
ENTER FIRST DATE:  1/1/1979
ENTER SECOND DATE: 4/1/1979
90 DAYS APART
```

8K ROM
1K RAM

Multisave

Doug Watson

Reliability of cassette storage and retrieval of programs and data requires multiple saves. Having to attend to the computer's needs as each save is completed is, however, a bit of a chore.

Wouldn't it be nice

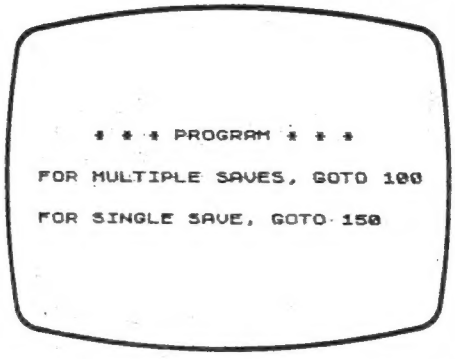
1) to key in the number of saves required and leave the ZX81 to complete that number of identical copies?

2) not have to check what variables have been assigned before incorporating this feature in a program?

3) to retain the normal, single-save mode?

If this appeals to you, try the demo program shown. The address shown as 16004 can be any unused, non-saved address in RAM (such as from 8192 to 16383 with the 64K Memopak or above RAMTOP) that is normally 0. PEEK it to find out.

```
  5 REM "PROGRAM NAME"
 10 PRINT AT 5,6;"* * * PROGRAM
* * *"
 20 PRINT "'''"  FOR MULTIPLE S
AVES, GOTO 100"
 30 PRINT "'''"  FOR SINGLE SAV
E, GOTO 150"
 40 STOP
100 PRINT "HOW MANY SAVES?"
110 PAUSE 40000
120 POKE 16507,VAL INKEY$-1
130 CLS
150 SAVE "PROGRAM NAME"
160 REM SLOW OR FAST HERE IF DE
SIRED
170 IF PEEK 16507=0 THEN RUN
180 POKE 16507,PEEK 16507-1
190 GOTO 150
```



On GOTO 100, HOW MANY SAVES? appears on the screen and waits (no L cursor). When one of the keys, 1 to 9, is pressed, the selected number of saves will be executed. If 0 is pressed, 256 saves will follow. Use BREAK to terminate.

On GOTO 150, a single save is executed.

No variables are used so compatibility with any program is automatic.

Tidying Up Your Display

Nick Godwin

A list of numbers is conventionally written down aligned on the right. This poses a problem on the ZX81, which lines numbers up the left.

Handwritten	ZX81 display
96	96
920	920
4	4
1020	1020

A routine to print, display, and summate a series of values may look as follows:

```
110 LET T=0
120 FOR J=1 TO 10
130 INPUT A
140 LET T=T+A
150 PRINT A
160 NEXT J
170 PRINT
180 PRINT T
999 STOP
```

The problem is, how to change the ZX81 display to conform to normal convention—how to align the values to the right.

The following amendment to the above does the trick, for integers (positive or negative) only:

```
110 LET T=0
120 FOR J=1 TO 10
130 INPUT A
140 LET T=T+A
145 LET X$=STR$ A
150 GOSUB 1000
160 NEXT J
170 PRINT
175 LET X$=STR$ T
180 GOSUB 1000
999 STOP
1010 PRINT TAB 17-LEN X$;X$
1020 RETURN
```

The routine prints the values neatly in the middle of the screen. However, it is very likely that you may want to print out decimal values in a similar format. The following subroutine (replacing the one above at line 1010) does the trick:

```
1010 IF X$(1)="." THEN LET X$="0
"+X$
1020 IF LEN X$>1 THEN IF X$(1 TO
 2)="-." THEN LET X$="-0"+X$(2 T
O )
1030 FOR K=1 TO LEN X$
1040 IF X$(K)="." THEN GOTO 1070
1050 NEXT K
1060 LET X$=X$+".0"
1070 PRINT TAB 17-K;X$
1080 RETURN
```

To test this routine, RUN the program and enter a series of wildly different values, e.g.:

46
9.009
-34456789.98765
-0.0000002
.0000034
8.6
88888.97
9999999
.01
.09

In practice, you will be fairly unlikely to require such a wide range of values for addition purposes, but the routine works equally well for any kind of arithmetical manipulation, or for any other list of numbers.

It may well be that you require the program to add sums of money, in which case you will want displayed only the value to two decimal places (representing pence or cents). To achieve this, the routine can be amended by adding or changing the following lines:

```
110 LET T=0
120 FOR J=1 TO 10
130 INPUT A
140 LET T=T+A
145 LET X=A
150 GOSUB 1000
160 NEXT J
170 PRINT
175 LET X=T
180 GOSUB 1000
999 STOP
1000 LET X$=STR$ (INT (X*100+.5)
/100)
1010 IF X$(1)="." THEN LET X$="0
"+X$
1020 IF LEN X$>1 THEN IF X$(1 TO
 2)="-." THEN LET X$="-0"+X$(2 T
O )
1030 FOR K=1 TO LEN X$
1040 IF X$(K)="." THEN GOTO 1070
1050 NEXT K
1060 LET X$=X$+".00"
1070 IF K=LEN X$-1 THEN LET X$=X
$+"0"
1080 PRINT TAB 17-K;X$
1090 RETURN
```

This routine prints the values neatly displayed in the middle of the screen, with the decimal points at TAB 16. The display routine does not affect the arithmetical calculation: you can enter as many digits as you like after the decimal point, and for the display they will be rounded to the first two decimal places. The lateral position of the display can be altered by adjusting the number (17) in line 1080, but care must be taken to allow for extra long numbers. The bug is that a value of .005 (half a pence or cent) prints as 0.00, not very serious, and probably not worth the bytes required to correct it.

The program as written occupies 394 bytes of RAM, 486 including the variables, and approximately 890 bytes including the display. Thus it is operational on 1K. The subroutine alone occupies 268 bytes.

Making Backups for Machine Language Tapes

Jack Ryan

Not long ago I bought a cassette of utility routines for my 16K ZX81. Although the routines on the tape were just what I needed, there was one problem: the program was in machine language and once LOADed above RAMTOP it ignored my SAVE commands. If something happened to that tape I was out of luck since I had no backup — unless I could make one by some method other than SAVE.

There are two methods which can be used to make a backup.

1. Copying to a Second Tape

The first technique is to simply copy the tape onto a second tape. This requires either a tape copier or a second tape recorder.

Place the original tape in Recorder 1 and a blank tape in Recorder 2. Use one of the Sinclair recorder leads to connect the earphone (output) jack of Recorder 1 to the microphone (input) jack of Recorder 2. Set Recorder 2 to RECORD and PAUSE, and (if it has one) the automatic level control (ALC) to OFF. Set the volume controls of each recorder to midrange. Start RECORDing on the second, then PLAYing on the first. After the recording is complete, try LOADing the program as usual with your recorder. Very likely you will have to try several different volume settings on your copying set up before getting a LOADable copy. A recording level meter on Recorder 2 helps. I found that I could not get a satisfactory recording without the ALC of Recorder 2 OFF.

2. Copying with PEEK and POKE

The second technique uses the PEEK and POKE facilities of the ZX81. The program to do this is Listing 1. Like the machine language routines that you see published, the software that I purchased is LOADed into memory above RAMTOP. RAMTOP is the address of the first nonexistent byte at the upper end of memory, which means that you cannot store anything at or above it—at least that is what the computer thinks. However, the address of RAMTOP is stored in the two bytes with addresses of 16388 and 16389. You can change these two bytes with POKE, and fool the computer into thinking that some existing memory is nonexistent. This memory is hidden from Basic and so is protected for machine language use.

PRINT PEEK 16388 − 256 * PEEK will tell you the address of RAMTOP, which is normally 32768 with the 16K RAM attached. The bytes stored at 16388 and 16389 (found by PEEKing each location) are 0 and 128. Note that 0 + 256 * 128 = 32768. Before loading a machine language tape, one or both of these bytes are changed by POKEing in new values to give RAMTOP a new, lower, address to correspond to the number of bytes required for the machine language routine. For example, the software I purchased comes with instructions to POKE 16389,124. So RAMTOP is now 0 + 256 * 124 = 31744, a difference of 32768 − 31744 = 1024 bytes. If you had a program with instructions to POKE 16388,100 and POKE 16389,125, then the new address of RAMTOP is 100 + 256 * 125 = 32100, and the number of bytes protected is 32768 − 32100 = 668.

Listing 1.

```
  40 FAST
  50 PRINT TAB 7;"ML PROGRAM BAC
KUP"
  60 PRINT
  70 REM "ENTER PARAMETERS FOR Y
OUR ML PROGRAM"
  80 PRINT "ENTER THE NUMBER TO
BE POKED"
  90 PRINT TAB 10;"INTO 16388";
 100 PRINT TAB 7;"IF NONE, ENTER
 0"
 110 INPUT A
 120 CLS
 130 PRINT "ENTER THE NUMBER TO
BE POKED"
 140 PRINT TAB 10;"INTO 16389";
 150 PRINT TAB 7;"IF NONE, ENTER
 0"
 160 INPUT B
 170 CLS
 180 PRINT "ENTER YOUR NORMAL RA
MTOP"
 190 PRINT "(17408 FOR 1K, 32768
 FOR 16K)"
 200 INPUT R
 210 CLS
 220 REM "C = NUMBER OF BYTES PR
OTECTED"
 230 LET C=R-(A+256*B)
 240 REM "D = LAST BYTE BEFORE N
EW RAMTOP"
 250 LET D=(A+256*B)-1
 260 REM "STORE MACHINE LANGUAGE
 IN E"
 270 DIM E(C)
 280 FOR N=1 TO C
 290 LET E(N)=PEEK (D+N)
 300 NEXT N
 310 STOP
 320 REM "PUT ML ABOVE RAMTOP"
 330 FOR  N=1 TO C
 340 POKE (D+N),E(N)
 350 NEXT N
 360 STOP
 370 SAVE "P"
 380 GOTO 0330
```

To make a copy of your machine language software, LOAD the program according to the instructions which accompany it. Then ENTER or LOAD the program in Listing 1. RUN the program and ENTER the requested data. Because the program runs in FAST, the screen blanks out while the program PEEKs the machine code into the variable array E. When the program stops with 9/310, prepare to SAVE the program. To SAVE, enter GOTO after you have begun recording. When you LOAD this SAVEd tape (LOAD "P") after altering RAMTOP as your machine language program requires, it is self-RUNning so the screen will blank out while the code is POKEd in above RAMTOP.

If you want to SAVE a copy of Listing 1 without any machine language code so that you can LOAD it in for a variety of machine language routines, enter SAVE "P", not GOTO 370.

The program as shown in Listing 1 is meant to be self documenting and user-friendly. But this is not necessary for operation of the program. Since this probably makes the program too large for a 1K machine, and makes it take longer to LOAD in any case, you might want to shorten it a bit. You can easily calculate the new address of RAMTOP and the number of bytes protected. So you could easily eliminate all REMs and replace lines 10 through 260 with

10 FAST
20 LET C = (calculated bytes protected)
30 LET D = (calculated new RAMTOP) − 1

In summary, you can make back up tapes for machine language routines which are LOADed above RAMTOP by copying your original tape from one recorder to a tape in a second recorder, or you can use a single recorder and the Basic program in Listing 1 to PEEK out the code for SAVEing and POKE it back in when LOADing from your back-up tape.